

# Aprendizaje Automático

## Máquinas de Vectores Soporte

curso 2018-2019

Enrique Vidal, Jorge Civera, Francisco Casacuberta

Departamento de Sistemas Informáticos y Computación

Universitat Politècnica de València

## 1. Introducción

El objetivo de esta práctica es experimentar con las máquinas de vectores soporte, o *Support Vector Machine* (SVM), en la plataforma `Octave`. Para ello se utiliza la librería `LibSVM` que se puede descargar de:

<http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

Para poder emplear dicha librería se deben tener compilados los ejecutables: `svm-train` y `svm-predict`. Para usarla desde `Octave` se necesitan tener los ficheros de código `Octave` `svmtrain.mex` y `svmpredict.mex`. Todo ello en el directorio de trabajo.

Todos estos requerimientos se han tenido en cuenta en la elaboración del toolkit `apr_toolkit`, que está en el mismo directorio de esta práctica. Para la realización de esta práctica se aconseja copiar la carpeta `apr_toolkit` a un directorio de trabajo del usuario en el que se realizarán todos los ejercicios.

## 2. Ejemplo de entrenamiento y clasificación con SVM

Según la teoría de SVM, dado un conjunto de aprendizaje  $S$ , formado por  $N$  vectores y sus correspondientes etiquetas de clase, el sistema de aprendizaje obtiene los *multiplicadores de Lagrange*,  $\alpha_n$  óptimos asociados a cada vector  $x_n$  de  $S$ ,  $1 \leq n \leq N$ . Los multiplicadores no nulos definen los vectores de  $S$  que constituyen el *soporte* de la función discriminante lineal del clasificador (en dos clases, por el momento). Concretamente, a partir de los multiplicadores no-nulos de los correspondientes vectores soporte y de sus etiquetas de clase, se obtienen fácilmente el vector de pesos  $\theta$  y el término independiente o *umbral*,  $\theta_0$ , que definen la función discriminante lineal del clasificador aprendido.

Por tanto, una vez obtenidos los multiplicadores (no-nulos), se puede construir de forma trivial un clasificador que prediga la etiqueta de clase asociada a cualquier vector (de prueba o “test”).

LibSVM, implementa el proceso de aprendizaje mediante `svm-train` y `svmtrain.mex`, y la predicción mediante `svm-predict` y `svmpredict.mex`. Un detalle importante de esta librería es que los multiplicadores obtenidos por los programas de aprendizaje (y los que usan los programas de predicción) pueden ser positivos o negativos. Se asume que los positivos corresponden a vectores soporte de la clase  $+1$  y los negativos de la  $-1$ . A todos los efectos, cualquiera de estos multiplicadores (que el toolkit denota como `sv_coef`), puede considerarse como el producto del verdadero multiplicador de Lagrange, por la etiqueta de clase del vector soporte correspondiente; es decir:  $c_n \alpha_n$ .

## 2.1. Corpus de ejemplo

Para introducir la funcionalidad básica de la librería LibSVM en Octave se propone emplear el corpus de `hart` que se encuentra en el directorio `data`. Es un corpus de dos clases donde los datos se representan en dos dimensiones ( $\mathbb{R}^2$ ). Este conjunto de datos *no* es linealmente separable en su espacio de representación original, pero sí lo es en un espacio transformado mediante un *núcleo* (“*kernel*”) de *base radial* (RBF), cuya dimensionalidad es mucho mayor. Los pasos a seguir podrían ser los siguientes.

Primero ejecutamos Octave en el directorio `apr_toolkit`. Una vez en Octave, cargamos los ficheros con los datos y las etiquetas de clase tanto del corpus de entrenamiento como del de prueba:

```
octave:1> load data/hart/tr.dat ; load data/hart/trlabels.dat
```

Una vez cargados los datos se podemos visualizarlos mediante:

```
octave:2> plot (tr(:,1),tr(:,2),"x")
```

Si se examinan `trlabels.dat` y `tslabels.dat`, puede observarse que las etiquetas de clase son “1” y “2”. LibSVM admite más de dos clases, que pueden denotarse mediante etiquetas numéricas (enteros positivos) cualesquiera. En el caso de dos clases, estas etiquetas se convierten internamente en  $+1$  y  $-1$  y esto se refleja en los signos de los multiplicadores (`sv_coef`), como se ha explicado anteriormente.

Para poder distinguir los vectores de cada clase se pueden dibujar los de la clase 1 y los de la clase 2 de manera separada:

```
octave:3> plot(tr(trlabels==1,1),tr(trlabels==1,2),"x",tr(trlabels==2,1),
tr(trlabels==2,2),"s")
```

Se observa cómo estos datos *no* son linealmente separables (en su espacio original,  $\mathbb{R}^2$ ).

## 2.2. Entrenamiento

Una breve ayuda para el uso de `svmtrain` se obtiene invocándolo sin argumentos:

```
octave:4> svmtrain
Usage: model = svmtrain(training_label_vector,
                        training_instance_matrix, 'libsvm_options');
libsvm_options:
[...]
```

Como resultado podemos ver los diferentes parámetros que podemos usar para el aprendizaje: tipo de kernel, parámetro  $C$  (para conjuntos de entrenamiento no-separables), grado del kernel si es polinomial, etc. Realizaremos un entrenamiento con kernel tipo RBF (`-t 2`):

```
octave:5> res = svmtrain(trlabels, tr, '-t 2 -c 1');
.*.*
optimization finished, #iter = 2298
nu = 0.174579
obj = -108.403744, rho = -0.096434
nSV = 398, nBSV = 91
Total nSV = 398
```

El resultado del proceso de entrenamiento se ha almacenado en la variable `res`, que es un tipo estructurado que contiene, entre otros: los parámetros del modelo, los índices de los vectores que han resultado ser vectores soporte, el multiplicador de Lagrange (multiplicado por la correspondiente etiqueta de clase,  $+1$  o  $-1$ ) asociado a cada vector soporte, etc. El contenido completo de `res` puede visualizarse mediante:

```
octave:6> res
```

Se puede acceder a cada uno de los campos de `res` es con el operador `."`. Por ejemplo se pueden mostrar los índices de los vectores soporte mediante:

```
octave:7> res.sv_indices
```

Y para mostrar los multiplicadores de Lagrange (multiplicados por las correspondientes etiquetas de clase,  $+1$  o  $-1$ ):

```
octave:8> res.sv_coef
```

**Ejercicio propuesto:** Representa gráficamente los vectores soporte, superponiéndolos a las muestras de entrenamiento.

## 2.3. Clasificación y estimación de la tasa de acierto

A partir del modelo entrenado en la etapa anterior (almacenado en la variable `res`), se pueden clasificar los vectores de un conjunto de prueba, que hay que cargar previamente:

```
octave:9> load data/hart/ts.dat ; load data/hart/tslabels.dat
```

La clasificación se realiza mediante la orden `svmpredict`. Como en el caso de `svmtrain`, para ver todas las opciones de `svmpredict` basta invocar `svmpredict` sin argumentos. Tanto en `svmtrain` como en `svmpredict` podremos poner entre `' '` cualquiera de las opciones que admite la librería LibSVM.

```
octave:10> svmpredict(tslabels,ts, res,' ');
Accuracy = 98.1% (981/1000) (classification)
```

El programa `svmtrain` *no* calcula los intervalos de confianza. Por tanto, estos intervalos los deberemos calcular nosotros como se ha visto en la practica 0.

### 3. Un pequeño ejercicio completo de aprendizaje (a entregar)

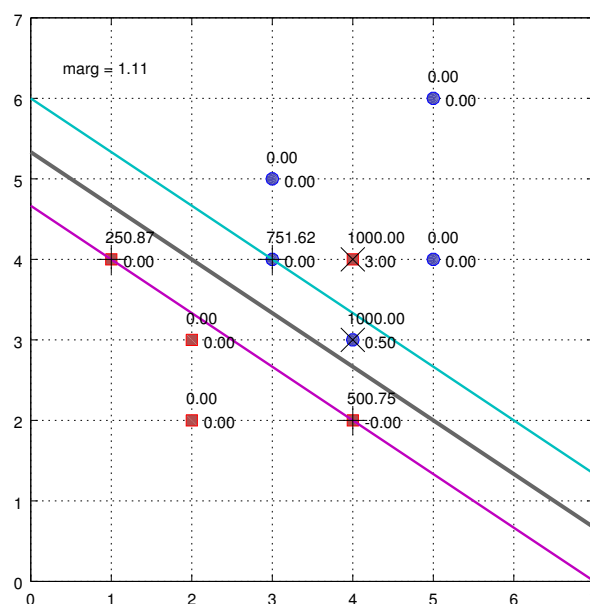
En el subdirectorio `data/mini` se encuentran dos pequeños conjuntos de datos de entrenamiento en dos dimensiones: (`trSep.dat`, `trSeplabels.dat`) y (`tr.dat`, `trlabels.dat`). El primero es linealmente separable (sin *kernel*) y el segundo no. Para cada uno de estos conjuntos:

1. Obtener una SVM sin kernel (es decir, kernel tipo lineal). Para simular la optimización estándar del caso separable basta usar un valor grande de  $\mathcal{C}$ ;
2. Determinar a) los multiplicadores de Lagrange,  $\alpha$ , asociados a cada dato de entrenamiento, b) los vectores soporte, c) el vector de pesos y umbral de la función discriminante lineal, y d) el margen correspondiente;
3. Calcular los parámetros de la frontera lineal (recta) de separación;
4. Representar gráficamente los vectores de entrenamiento, marcando los que son vectores soporte, y la recta separadora correspondiente.

Además, para el conjunto no-separable:

- 2.1 Determinar los valores de tolerancia de margen,  $\zeta$ , asociados a cada dato de entrenamiento;
- 4.1 En la representación gráfica, marcar los vectores soporte “erróneos”;
- 4.2 Obtener los resultados (gráficas) indicados anteriormente para diversos valores relevantes de  $\mathcal{C}$ .

Un ejemplo “ideal” de representación gráfica para el caso no-separable, obtenido con  $\mathcal{C} = 1000$ , se muestra en la siguiente figura:



## 4. Ejercicios con un corpus de dos clases

Se propone la realización de un experimento de clasificación completo: entrenamiento y evaluación. Para ello se empleará un corpus, de dos clases para detección de *Spam* compuesto por 3220 muestras de aprendizaje y 1381 de test. Los documentos se representan mediante vectores de 58 componentes, que son proporcionales a las frecuencias de uso de un conjunto de 58 palabras clave. Los ficheros de entrenamiento y test se encuentran en el directorio `data/spam`.

Se pide determinar los parámetros del modelo, concretamente el tipo de kernel y el valor de  $\mathcal{C}$ , para los que se obtienen mejores resultados. Se deberá presentar una tabla y/o gráfica donde se muestre no solo el mejor resultado obtenido, sino también otros resultados relevantes que permitan poner de manifiesto la tendencia a mejorar o empeorar del modelo según varían el o los parámetros considerados.

Recordad que toda estimación de la probabilidad de error o de acierto de un clasificador, debe ir acompañada de sus correspondientes intervalos de confianza (al 95 %).

## 5. Ejercicios con un corpus multi-clase

Se propone la realización de un experimento de clasificación completo para un corpus de 10 clases (USPS) para reconocimiento de dígitos manuscritos. Está compuesto por 7291 muestras de aprendizaje y 2007 de test. Las dígitos se representan mediante vectores de 256 componentes, proporcionales a los valores de gris de imágenes de  $16 \times 16$  píxeles. Los ficheros de entrenamiento y test se encuentran en el directorio `usps` de `data`. Se pide:

- a) Determinar los parámetros del modelo, concretamente el tipo de kernel y el valor de  $\mathcal{C}$ , para los que se obtienen mejores resultados. Se deberá presentar una tabla y/o gráfica donde se muestre no solo el mejor resultado obtenido, sino también otros resultados relevantes que permitan poner de manifiesto la tendencia a mejorar o empeorar del modelo según varían el o los parámetros considerados.

Recordad que toda estimación de la probabilidad de error o de acierto de un clasificador, debe ir acompañada de sus correspondientes intervalos de confianza (al 95 %).

- b) Visualizar algunas de las imágenes de aprendizaje y algunos de los vectores (imágenes) soporte obtenidos en el mejor clasificador. Para ello pueden ser de ayuda los comandos de Octave `‘‘imshow’’` y `‘‘reshape’’`.